Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis

Thi-Bich-Hanh Dao

▶ **To cite this version:**

Thi-Bich-Hanh Dao. Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis. JFPLC, 2000, Marseille, France. pp.225-240. hal-00144920

HAL Id: hal-00144920
https://hal.science/hal-00144920

Submitted on 12 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis

Dao Thi-Bich-Hanh

Laboratoire d'Informatique de Marseille
Parc Scientifique et Technologique de Luminy
163, avenue de Luminy - Case 901, 13288 Marseille Cedex 9, France
e-mail : dao@lim.univ-mrs.fr

RÉSUMÉ. *Nous présentons dans ce papier un algorithme de résolution, dans la théorie* ***T*** *des arbres (éventuellement infinis), des contraintes représentées par des formules générales du premier ordre, avec pour seul symbole de relation l'égalité et pour symboles de fonctions les éléments d'un ensemble infini* ***F****. L'algorithme est constitué d'un ensemble de 11 règles de réécriture de sous-formules en formules équivalentes. Il transforme une formule du premier ordre en une conjonction de formules « résolues », équivalente dans* ***T****, ne faisant pas intervenir de nouvelles variables libres et ayant les deux propriétés qui suivent. (1) La conjonction de formules résolues, soit est la constante logique vrai, soit se réduit à la proposition ¬ vrai, soit a au moins une variable libre et n'est équivalente ni à vrai ni à faux dans la théorie* ***T****. (2) Chaque formule résolue se transforme d'une façon immédiate en une combinaison booléenne de formules de bases dont la longueur n'excède pas le double de celle de la formule résolue. Les formules de bases sont des cas particuliers de conjonctions d'équations avec quantifications existentielles. La correction de l'algorithme constitue une autre preuve de la complétude de* ***T*** *démontrée par Michael Maher. Nous terminons avec quelques benchmarks effectués par une implantation de l'algorithme, résolvant des formules qui font intervenir des imbrications de plus de 160 quantificateurs alternés.*

ABSTRACT. *We present in this paper an algorithm, in the theory* ***T*** *of (eventually infinite) trees, for solving constraints represented by full first order formulae, with equality as the only relation and with symbols of function taken in an infinite set* ***F****. The algorithm consists of a set of 11 rewrite rules. It transforms a first order formula in a conjunction of "solved" formulae, equivalent in* ***T****, which has not new free variables and which is such that, (1) the conjunction either is the constant logic true or is reduced to ¬ true, or has at least one free variable and is equivalent neither to true nor to false, (2) each solved formula can be transformed immediately in a Boolean combination of basic formulae whose length does not exceed twice the length of the solved formula. The basic formulae are particular cases of existentially quantified conjunctions of equations. The correctness of the algorithm gives another proof of the completeness of* ***T*** *demonstrated by Michael Maher. We end with benchmarks realized by an implementation, solving formulae with more than 160 nested alternated quantifiers.*

MOTS-CLÉS : *Théorie des arbres infinis, Formule du premier ordre, Règle de réécriture, Contrainte*

KEYWORDS: *Theory of infinite trees, First order formula, Rewrite rule, Constraint*

2e soumission à *JFPLC'2000*, le 5 juin 2000.

1. Introduction

Les arbres éventuellement infinis, jouent un rôle fondamental en informatique. Ils modélisent aussi bien des structures de données que des schémas ou des déroulements de programme. Dès 1976, Gérard Huet a proposé un algorithme pour unifier des termes infinis, c'est-à-dire de résoudre des équations dans les arbres [HUE 76]. Bruno Courcelle a étudié les propriétés des arbres infinis notamment dans le cadre des schémas récursif de programme [COU 83, COU 86]. Alain Colmerauer a modélisé le fonctionnement de Prolog II, III et IV par la résolution d'équations et de diséquations dans les arbres infinis [COL 82, COL 83, COL 84, COL 90, BEN 96]. Il est donc vital de savoir résoudre des contraintes générales du premier ordre avec une seule relation d'égalité dans les arbres. Michael Maher a proposé et justifié une théorie complète **T** des arbres finis ou infinis sur un ensemble **F** (fini ou infini) de symboles [MAH 88]. Entre autres, il montre que pour toute formule p, il existe une formule q, équivalente à p dans cette théorie, qui est une combinaison booléenne de conjonctions d'équations quantifiées existentiellement. Si la formule p ne contient pas d'occurrences libres de variables, alors la formule q est soit *vrai* soit *faux*. Le cas où **F** est fini est également considéré dans un travail de Hubert Comon [COM 88].

Pour notre part, nous avons isolé un ensemble de 11 règles de réécriture de sous-formules, qui réalisent la résolution de contraintes générales du premier ordre dans la théorie **T** avec l'ensemble **F** infini. Chaque stratégie d'application de ces règles correspond à un algorithme de résolution. La formule finale, équivalente dans **T** à la formule initiale qui contient éventuellement des occurrences libres de variables, soit est *vrai*, soit est $\neg$ *vrai*, soit contient au moins une variable libre et n'est équivalente ni à *vrai* ni à *faux*. Nous présentons cette résolution dans l'ordre suivant.

Dans la section 2, nous rappelons la théorie **T** et montrons des propriétés la concernant. Dans la section 3, nous présentons les formes de formules. La section 4 est consacrée à la présentation de l'algorithme, effectué principalement par un système de 11 règles de réécriture de sous-formules, et à la correction de cet algorithme. Nous présentons dans la section 5 quelques exemples et benchmarks réalisés par l'implantation de cet algorithme et concluons dans la section 6.

2. La théorie des arbres finis ou infinis

2.1. *Syntaxe*

Soit **V** un ensemble infini dénombrable dont les éléments sont appelés *variables* et **F** un ensemble infini dont les éléments sont appelés *symboles fonctionnels* et à chacun desquels est associé un entier n positif ou nul, son *arité*. On suppose que l'ensemble **V** est ordonné par une relation d'ordre $\succeq$. On écrit $u \succ v$ pour signifier que $u \succeq v$ et $u \neq v$ et on suppose que l'ordre $\succ$ soit dense. Les *formules* sont des expressions de l'une des douze formes :

$$s = t, \textit{vrai}, \textit{faux}, \neg(p), (p \wedge q), (p \vee q), (p \rightarrow q), (p \leftrightarrow q), \exists x\, p, \forall x\, p, \exists ?x\, p, \exists !x\, p,$$

où p et q sont des formules de longueurs plus courtes, x est un vecteur construit sur **V** et s, t des *termes*, c'est-à-dire des expressions de l'une des deux formes :

$$u, ft_1 \ldots t_n$$

avec u une variable, $n \geq 0$, $f \in$ **F** d'arité n et les t_i des termes de longueurs plus courtes. Les formules $\exists ?x\ p$ et $\exists !x\ p$ signifient informellement « il existe au plus une valeur pour chaque variable de x, telle que p » et « il existe une et une seule valeur pour chaque variable de x, telle que p ». Une occurrence de la variable x_i dans une formule est *liée* ou *libre* suivant qu'elle se produit ou pas à l'intérieur d'une sous-formule de l'une des quatre dernières formes, avec x_i figurant dans x. Les *variables libres* d'une formule p sont les éléments de **V** qui ont au moins une occurrence libre dans p. Une *proposition* est une formule sans variables libres.

Une *algèbre* est constituée d'un ensemble non vide A et d'une fonction $\hat{f} : A^n \to A$ pour chaque symbole fonctionnel f d'arité n. Une *théorie* T est un ensemble (éventuellement vide ou infini) de propositions, appelées les *axiomes* de la théorie. A la façon de Roger Lyndon [LYN 64], nous voyons une *interprétation* comme une fonction qui associe à chaque variable un élément d'un domaine A, à chaque symbole de fonction d'arité n une fonction $A^n \to A$ et, à chaque formule une valeur de vérité *vrai* ou *faux*, calculée en attribuant aux symboles logiques leur signification usuelle. Une interprétation *satisfait* T si elle interprète toute proposition de T par *vrai*. On dit que la formule p *est vraie dans* T et on écrit $T \models p$ si p est vraie dans toute interprétation satisfaisant T. Remarquons que si $x_1, \ldots, x_n$ sont les variables libres de p, alors $T \models p$ est équivalent à $T \models \forall x_1 \ldots x_n\ p$. On dit que les formules p et q sont *équivalentes dans la théorie* T ssi $T \models p \leftrightarrow q$.

Terminons par trois propriétés concernant des formules quelconques p, q, une formule $r[p]$ ayant une occurrence de p et la formule $r[q]$ obtenue en remplaçant cette occurrence par q.

Propriété 1 $T \models p \leftrightarrow q$ *entraîne* $T \models r[p] \leftrightarrow r[q]$.

Propriété 2 $T \models \exists ?x\ p$ *entraîne* $T \models (\exists x\ p \wedge \neg q) \leftrightarrow (\exists x\ p) \wedge \neg(\exists x\ p \wedge q)$.

Propriété 3 $T \models \exists !x\ p$ *entraîne* $T \models (\exists x\ p \wedge \neg q) \leftrightarrow \neg(\exists x\ p \wedge q)$.

2.2. *Les axiomes*

On suppose à partir de maintenant que l'ensemble **F** de symboles fonctionnels est infini. La théorie **T** des arbres finis ou infinis (sur **F**) a pour axiomes l'ensemble infini des propositions de l'une des trois formes :

$$\forall x \forall y \qquad \neg fx = gy \qquad [1]$$

$$\forall x \forall y \qquad fx = gy \ \rightarrow \ x = y \qquad [2]$$

$$\forall x \exists ! y \qquad y = t[y, x] \qquad [3]$$

où f et g sont des symboles fonctionnels distincts et le vecteur $t[y, x]$ est composé de termes $t_i[y, x]$ faisant intervenir au moins un symbole fonctionnel et pas d'autres variables que celles figurant dans yx.

Michael Maher [MAH 88] a introduit cette théorie et a montré qu'elle était complète, c'est-à-dire que, quelle que soit la proposition p, on a soit $\mathbf{T} \models p$, soit $\mathbf{T} \models \neg p$. Ainsi que nous l'avons déjà mentionné, il a aussi montré que pour toute formule p, il existe une formule q, équivalente à p dans **T**, qui est une combinaison booléenne de conjonctions d'équations quantifiées existentiellement.

2.3. *L'algèbre des arbres finis ou infinis*

On peut montrer que l'ensemble des propriétés du premier ordre qui constitue la théorie **T** est un sous-ensemble de l'ensemble des propriétés du premier ordre de *l'algèbre* $(A, \boldsymbol{F})$ *des arbres finis ou infinis*, où informellement,

– A est l'ensemble des arbres ayant un ensemble éventuellement infini de nœuds, chacun d'eux ayant un nombre fini de fils n, et étant étiqueté par un élément de **F** d'arité n,

– chaque élément f de **F**, d'arité n, est interprété comme l'opération $(a_1, \ldots, a_n) \mapsto a_0$, où a_0 est l'arbre dont la racine est étiquetée f et dont la suite de fils est $a_1, \ldots, a_n$.

De ce fait et du fait que **T** est une théorie complète, pour toute proposition p on aura $\mathbf{T} \models p$ ou $\mathbf{T} \models \neg p$, suivant que p est ou n'est pas une propriété de l'algèbre $(A, \mathbf{F})$. La théorie **T** axiomatise donc l'algèbre des arbres finis ou infinis.

2.4. *Propriétés des conjonctions d'équations*

On appelle *équation élémentaire* une formule de la forme $u = t$, où u est une variable et t un terme *à plat*, c'est-à-dire de la forme v ou $fv_1 \ldots v_n$, avec v et v_i des variables et f un symbole fonctionnel d'arité n. Une *conjonction d'équations élémentaires* est une formule de la forme $u_1 = t_1 \wedge \cdots \wedge u_n = t_n \wedge$ *vrai*, avec $n \geq 0$ et les $u_i = t_i$ des équations élémentaires. Une conjonction d'équations élémentaires est *résolue* si tous ses membres gauches sont distincts et si aucune de ses équations n'est de la forme $u = v$, où u et v sont des variables telles que $v \succeq u$.

Dans la formule $(\exists x \bigwedge_{i=1}^{n} v_i = t_i)$, où les $v_i = t_i$ sont des équations élémentaires, les équations et les variables *accessibles* sont celles qui figurent dans au moins une de ses sous-formules de la forme $\bigwedge_{j=1}^{m} v_{k_j} = t_{k_j}$, avec v_{k_1} ne figurant pas dans x et $v_{k_{j+1}}$ figurant dans t_{k_j} pour tout $j \in 1..(m-1)$. On montre les propriétés suivantes :

Propriété 4 *Si a est une conjonction d'équations élémentaires et si toutes les variables et toutes les équations de $\exists x\, a$ sont accessibles, alors $\boldsymbol{T} \models \exists ?x\, a$.*

Propriété 5 *[MAH 88] Si a est une conjonction résolue d'équations élémentaires et x est la suite de ses membres gauches, alors $\boldsymbol{T} \models \exists! x\ a$.*

Propriété 6 *[COL 84] Soient a et b des conjonctions résolues d'équations élémentaires. Si a et b ont les mêmes membres gauches et $\boldsymbol{T} \models a \rightarrow b$, alors $\boldsymbol{T} \models a \leftrightarrow b$.*

Propriété 7 *Soient les $\exists x_i\ a_i$ des conjonctions quantifiées d'équations élémentaires telles que toutes les variables de x_i sont accessibles et, soit x un vecteur de variables. Si chaque conjonction a_i contient une occurrence d'une variable u de x dans une équation dont la forme est différente de $u = u$ et de $u = v$, avec v figurant dans x_i, alors*

$$\boldsymbol{T} \models \exists x \bigwedge_{i=1}^{n} \neg(\exists x_i\ a_i) \qquad [4]$$

3. Formes de formules

Dans la section précédente, nous avons présenté les formes générales. Nous présentons maintenant la forme normalisée sur laquelle notre algorithme fonctionne, une classe de forme normalisée, appelée forme résolue et montrons leurs propriétés.

3.1. *Forme normalisée*

Définition 8 Une formule *normalisée* de profondeur $d \geq 1$ est de la forme

$$p = \neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} p_i), \qquad \text{avec } n \geq 0, \qquad [5]$$

où a est une conjonction d'équations élémentaires et les p_i des formules normalisées de profondeur d_i, avec $d = 1 + \max\{0, d_1, \ldots, d_n\}$. De plus, on impose la discipline suivante pour nommer les variables quantifiées :

– si v est une variable de x et u une variable libre de p, alors on a $v \succ u$,

– si v est une variable de x et w une variable de y_i, avec $p_i = \neg(\exists y_i \ldots)$, alors $w \succ v$, et

– les variables sont aussi distinctes que possible.

Il est facile de transformer une formule quelconque en une formule normalisée équivalente dans la théorie vide. Il suffit par exemple (1) d'introduire un supplément d'équations et de variables quantifiées existentiellement pour transformer les conjonctions d'équations en conjonctions d'équations élémentaires, (2) d'exprimer tous les quantificateurs, constantes et connecteurs logiques uniquement à l'aide de *vrai*, $\neg, \wedge, \exists$, (3) d'enlever les doubles négations ($\neg\neg p$ devient p), (4) si la formule p ainsi obtenue ne

commence pas par $\neg$, de la remplacer par $\neg\neg p$, (5) de nommer les variables quantifiées par des noms différents de ceux des variables libres et aussi différents que possible, (6) de faire passer les quantifications au dessus des conjonctions ($p \wedge (\exists v q)$ devient $\exists v p \wedge q$, car les variables libres de p sont distinctes de v). Si la formule de départ ne contient ni le connecteur $\leftrightarrow$, ni les quantificateurs $\exists ?, \exists !$, cette transformation se fait linéairement, c'est-à-dire qu'il existe une constante k telle que $n_2 \leq k n_1$, où n_1 est la taille de la formule à transformer et n_2 celle de la formule obtenue.

3.2. *Forme résolue*

Définition 9 Une formule *résolue* est une formule normalisée de la forme

$$\neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i\ b_i)), \qquad \text{avec } n \geq 0 \qquad [6]$$

où a et les éventuels b_i sont des conjonctions d'équations élémentaires, qui respectent les conditions suivantes :

1. Les conjonctions a et $a \wedge b_i$ sont résolues.
2. Toutes les variables et équations de $\exists x\ a$ sont accessibles et, pour chaque i, toutes les variables et équations de $\exists y_i\ b_i$ sont accessibles.
3. Aucun b_i n'est égal à *vrai.*

On montre que les formules résolues ont la propriété suivante :

Propriété 10 *Si p est une formule résolue et si $\boldsymbol{T} \models p$ alors $p =$ vrai.*

D'autre part, d'une façon immédiate, d'après les propriétés 4 et 2, toute formule résolue se transforme en une combinaison booléenne de conjonctions résolues d'équations élémentaires :

Propriété 11 *Dans **T**, toute formule résolue $\neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i\ b_i))$ est équivalente à la formule $\neg(\exists x\ a) \vee \bigvee_{i=1}^{n} (\exists x y_i\ a \wedge b_i)$.*

On annonce maintenant le théorème principal qui est le résultat de l'algorithme présenté dans ce papier.

Théorème 12 *Dans **T**, toute formule est équivalente soit à vrai, soit à faux, soit à une conjonction de formules résolues, qui a au moins une variable libre et qui n'est équivalente ni à vrai ni à faux.*

Par conséquence, si la formule initiale n'a pas de variables libres, elle est équivalente soit à *vrai* soit à *faux.* Ceci constitue une autre preuve de la complétude de la théorie **T**. De ce théorème et de la propriété 11, nous avons le corollaire suivant, à travers lequel nous retrouvons le résultat de Michael Maher.

Corollaire 13 *Dans **T**, toute formule est équivalente à une combinaison booléenne de conjonctions d'équations quantifiées existentiellement (partiellement ou totalement).*

4. Algorithme de résolution

La résolution de formules est réalisée par un système de 11 règles de réécriture de sous-formules. Ces règles de réécritures fonctionnent sur des formules de travail, qui sont des formules normalisées, où les occurrences de $\neg$ portent un numéro qui indique les propriétés de la formule. Nous présentons dans cette partie la forme de travail, l'ensemble des règles de réécriture, l'algorithme de résolution et la démonstration de la correction de l'algorithme.

4.1. *Formule de travail*

Définition 14 Une *formule de travail* est une formule normalisée dont les occurrences de $\neg$ ont été remplacées par des $\neg^k$, avec k pris dans $0..4$, et telle que toute occurrence de sous-formule de la forme

$$p = \neg^k (\exists x\ a \wedge q), \qquad \text{avec } k > 0, \tag{7}$$

satisfasse aux k premières conditions de la liste de conditions ci-dessous. Ici a est une conjonction d'équations, q une conjonction de formules de travail et, a' désigne la partie d'équations de la sur-formule de travail immédiate de p, si elle existe.

1. Si a' existe, alors $\mathbf{T} \models a \ \rightarrow \ a'$ et l'ensemble des membres gauches de a' est inclus dans celui de a.

2. Les membres gauches de la conjonction a sont tous distincts et a ne contient pas d'équation de la forme $u = v$, avec $v \succeq u$.

3. Si a' existe, l'ensemble des équations de a' est inclus dans celui de a.

4. Les équations de a et les variables de x sont toutes accessibles dans p et, si a' existe, chaque équation de a' figure dans a.

Qualifions d'*initiale* une formule de travail commençant par $\neg^3$ et telle que $k = 0$ pour toutes les autres occurrences de $\neg^k$. Qualifions de *finale* une formule de travail de profondeur inférieure ou égale à 2, avec $k = 4$ pour toutes les occurrences de $\neg^k$. La relation entre les formules de travail finales et les formules résolues s'énonce ainsi :

Propriété 15 *Soit la formule de travail finale* $p = \neg^4(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg^4(\exists y_i\ a \wedge b_i))$. *La formule* $\neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i\ b_i))$ *est une formule résolue, équivalente à* p*, dans la théorie vide.*

4.2. *Les 11 règles de réécriture*

Nous présentons maintenant les règles de réécriture pour transformer une conjonction de formules de travail de profondeur quelconque en une conjonction de formules de travail finales, équivalente dans $\mathbf{T}$. L'application d'une règle $p_1 \implies p_2$ à une conjonction p consiste à remplacer dans p une occurrence de la sous-formule p_1 par

la formule p_2 à condition que le résultat reste une formule de travail. Dans ce qui suit, les lettres u, v, w désignent des variables, les lettres x, y, z désignent des vecteurs de variables, les lettres a, b, c des conjonctions d'équations, la lettre p une formule de travail, la lettre q une conjonction de formules de travail, la lettre r une conjonction d'équations et de formules de travail. Ces lettres peuvent être altérées par des primes ou des indices. On suppose que le connecteur $\wedge$ est commutatif et associatif.

$$
\begin{array}{lll}
1 & \neg^1(\exists x\ u = u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ r) \\
2 & \neg^1(\exists x\ v = u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u = v \wedge r) \\
3 & \neg^1(\exists x\ u = v \wedge u = t \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u = v \wedge v = t \wedge r) \\
4 & \neg^1(\exists x\ u = f v_1..v_n \wedge u = g w_1..w_m \wedge r) & \Longrightarrow \quad \textit{vrai} \\
5 & \neg^1(\exists x\ u = f v_1..v_n \wedge u = f w_1..w_n \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u = f v_1..v_n \wedge \bigwedge_{i=1}^{n} v_i = w_i \wedge r) \\
6 & \neg^1(\exists x\ r) & \Longrightarrow \quad \neg^2(\exists x\ r)
\end{array}
$$

$$
\begin{array}{lll}
7 & \neg^3(\exists x\ a \wedge q \wedge \neg^0(\exists y\ r)) & \Longrightarrow \quad \neg^3(\exists x\ a \wedge q \wedge \neg^1(\exists y\ a \wedge r)) \\
8 & \neg^3(\exists x\ a \wedge q \wedge \neg^2(\exists y\ a' \wedge r)) & \Longrightarrow \quad \neg^3(\exists x\ a \wedge q \wedge \neg^3(\exists y\ a \wedge r)) \\
9 & \neg^3(\exists x\ a \wedge \neg^4(\exists y\ a) \wedge q) & \Longrightarrow \quad \textit{vrai} \\
10 & \neg^3 \left[\begin{array}{l} \exists x x'\ a \wedge a' \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists y_i y_i'\ b_i[x'] \wedge b_i') \end{array} \right] & \Longrightarrow \quad \neg^4 \left[\begin{array}{l} \exists x\ a \wedge \\ \bigwedge_{i \in K} \neg^4(\exists y_i x_i'\ b_i[x_i']) \end{array} \right] \\
11 & \neg^3 \left[\begin{array}{l} \exists x\ a \wedge q[x] \wedge \\ \neg^4 \left[\begin{array}{l} \exists y\ b \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists z_i\ c_i[xy]) \end{array} \right] \end{array} \right] & \Longrightarrow \quad \left[\begin{array}{l} \neg^3(\exists x\ a \wedge q[x] \wedge \neg^4(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^3(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i]) \end{array} \right]
\end{array}
$$

Dans ces règles, les variables u et v sont telles que $u \succ v$, les f, g sont des symboles fonctionnels distincts. Dans la règle 8, a et a' ont les mêmes membres gauches d'équations. Dans la règle 10, (x, a) est le couple de variables et d'équations accessibles de $\exists x x'\ a \wedge a'$ et, pour chaque $i \in 1..n$, $(y_i x_i', b_i)$ est le couple de variables et d'équations accessibles de $\exists y_i y_i' x''\ b_i \wedge b_i'$, avec x'' le vecteur des membres gauches de a'. L'ensemble K est un sous-ensemble de $1..n$ tel que $i \in K$ ssi les variables de x' ne figurant pas dans x'' ne figurent pas dans b_i. Dans la règle 11, dans $q[x_i]$, les occurrences de $\neg^k$ sont toutes remplacées par $\neg^0$.

Le résultat de la transformation d'une formule par des règles de réécriture se résume dans la propriété suivante, dont la preuve est exhibée dans la sous-section 4.4.

Propriété 16 *Toute application répétée de ces règles de réécriture sur une formule de travail initiale p se termine et produit une conjonction de formules finales, équivalente à p dans **T**.*

4.3. *Algorithme*

Etant donnée une formule p, la résolution de p est effectuée par les étapes suivants :

1. Transformation de p en une formule normalisée p_1, équivalente à p dans la théorie vide.

2. Transformation de p_1 en la formule de travail p_2

$$p_2 = \neg^3(\exists\varepsilon\ \mathit{vrai} \wedge \neg^0(\exists\varepsilon\ \mathit{vrai} \wedge p_1)), \qquad [8]$$

où ε est le vecteur vide sur $\mathbf{V}$ et où les occurrences de $\neg$ dans p_1 sont remplacées par des $\neg^0$.

3. Application des règles de réécriture sur p_2, autant de fois que possible. Celle-ci produit une conjonction de formules finales p_3 qui, d'après la propriété 15, correspond à une conjonction de formules résolues.

4.4. *Correction de l'algorithme*

Le théorème 12 provient des résultats suivants. Supposons que la propriété 16 soit correcte. D'après les définitions des règles de réécriture, la formule p_2 est transformée de la façon suivante, où q et p_3 sont des conjonctions de formules finales :

$$p_2 \Longrightarrow \neg^3(\exists\varepsilon\ \mathit{vrai} \wedge \neg^3(\exists\varepsilon\ \mathit{vrai} \wedge p_1)) \Longrightarrow^* \neg^3(\exists\varepsilon\ \mathit{vrai} \wedge q) \Longrightarrow^* p_3$$

Les formules p_2 et p_3 sont équivalentes dans $\mathbf{T}$. Si $\mathbf{T} \models p$, d'après la propriété 10, la formule p_3 et *vrai*. Si $\mathbf{T} \models \neg p$, alors $\mathbf{T} \models q$. Encore d'après la propriété 10, q est *vrai*. La formule p_3 est donc $\neg^4(\exists\varepsilon\ \mathit{vrai})$. Sinon, p_3 n'est équivalente ni à *vrai* ni à *faux* dans $\mathbf{T}$. Du fait que p_3 n'est pas *vrai*, elle contient des équations qui, d'après les conditions de l'indice 4 des formules de travail, doivent être accessibles. Par conséquence, la conjonction p_3 contient au moins une variable libre. Il ne nous reste plus qu'à démontrer la propriété 16, c'est-à-dire la terminaison et la correction des règles de réécriture.

4.4.1. *Terminaison des règles de réécriture*

Du fait que l'ensemble de variables $\mathbf{V}$ est ordonné par la relation d'ordre $\succeq$, on peut avoir une fonction qui associe à chaque variable un nombre entier non négatif distinct, son numéro, de telle façon que le numéro de la variable u soit plus grand que celui de la variable v ssi $u \succ v$. Soit p la formule de travail considérée. On introduit les nombres n_i entiers non négatifs, avec $i = 1..8$ comme suit :

– $n_1 = \alpha(p)$ où la fonction α est définie par :

$$\alpha(\mathit{vrai}) = 0,$$
$$\alpha(\neg(\exists x\, a \wedge p_1 \wedge \cdots \wedge p_n)) = 2^{\alpha(p_1)+\cdots+\alpha(p_n)},$$

il est clair que $\alpha(p_1) < \alpha(p_2)$ entraîne $\alpha(p[p_1]) < \alpha(p[p_2])$,

– n_2 le nombre d'occurrences de $\neg^0$,

– n_3 le nombre d'occurrences de $\neg^1$,

– n_4 le nombre d'occurrences de symboles fonctionnels dans les sous-formules $\neg^1(\ldots)$,

– n_5 la somme des numéros de variables dans les sous-formules $\neg^1(\ldots)$,

– n_6 le nombre d'équations de la forme $v = u$ avec $u \succ v$ dans les sous-formules $\neg^1(\ldots)$,

– n_7 le nombre d'occurrences de $\neg^2$,

– n_8 le nombre d'occurrences de $\neg^3$.

Pour chaque règle, il existe un indice i tel que l'application de la règle garde inchangés les n_j, avec $1 \leq j < i$, et diminue n_i. Ces i valent 1 pour les règles 4, 9 et 11, vaut 2 pour la règle 7, vaut 3 pour la règle 6, vaut 4 pour la règle 5, valent 5 pour les règles 1 et 3, vaut 6 pour la règle 2, vaut 7 pour la règle 8 et vaut 1 ou 8 pour la règle 10. Du fait que les n_i sont tous entiers non négatifs, ils ne peuvent pas être diminués infiniment. L'application des règles se termine donc.

4.4.2. *Correction des règles de réécriture*

On montre maintenant que pour une telle règle $p \Longrightarrow p'$, les formules p et p' sont équivalentes dans **T** et la formule p' reste une formule de travail.

4.4.2.1. Justification des règles 1-9

Les règles 1,2,3,6,7 et 9 sont correctes dans la théorie vide. Les règles 4 et 5 sont correctes d'après les deux premiers axiomes de **T**, respectivement. Dans la règle 8, par les propriétés de l'indice 2 d'une formule de travail, la formule p est de la forme :

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ a' \wedge b \wedge q'))$$

où $\mathbf{T} \models a' \wedge b \ \rightarrow\ a$. On a donc $\mathbf{T} \models a' \wedge b \ \rightarrow\ a \wedge b$. Puisque ces deux conjonctions ont les mêmes membres gauches d'équations, d'après la propriété 6, on a $\mathbf{T} \models a' \wedge b \leftrightarrow a \wedge b$, ce qui entraîne la correction de la règle.

4.4.2.2. Justification de la règle 10

$$\neg^3 \left[\begin{array}{l} \exists x x'\ a \wedge a' \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists y_i y_i'\ b_i[x'] \wedge b_i') \end{array} \right] \Longrightarrow \neg^4 \left[\begin{array}{l} \exists x\ a \wedge \\ \bigwedge_{i \in K} \neg^4(\exists y_i x_i'\ b_i[x_i']) \end{array} \right]$$

Ici les variables de x et les équations de a sont accessibles dans $(\exists x x'\ a \wedge a')$ et, pour chaque i, les variables de $y_i x_i'$ et les équations de b_i sont accessibles dans $(\exists y_i y_i' x''\ b_i \wedge b_i')$, avec x'' le vecteur des membres gauches de a'. Par conséquence, aucune variable de x' ne figure dans a et aucune variable de $y_i'(x'' - x_i')$ ne figure dans b_i. D'après

la propriété 5, on a $\mathbf{T} \models \exists! x'' \, a'$ et, d'après la propriété 3, la formule gauche est équivalente dans **T** à la formule suivante :

$$\neg(\exists x(x' - x'') \; a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i y_i' x'' \; a' \wedge b_i \wedge b_i'))$$

D'après la définition de la formule de travail, l'ensemble des équations de a' est inclus dans celui de $b_i \wedge b_i'$. En faisant descendre les variables inaccessibles, la formule devient :

$$\neg(\exists x(x' - x'') \; a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i x_i' \; b_i \wedge (\exists y_i'(x'' - x_i') \; b_i')))$$

où $x_i' \subseteq x''$ et aucune variable de $(x'' - x_i')$ ne figure dans b_i. Puisqu'aucune équation de b_i' n'est accessible dans $(\exists y_i y_i' x'' \; b_i \wedge b_i')$, les membres gauches de b_i' figurent tous dans $y_i'(x'' - x_i')$. D'après la propriété 5, $\mathbf{T} \models \exists! y_i'(x'' - x_i') \; b_i'$. La formule précédente est équivalente à :

$$\neg(\exists x(x' - x'') \; a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i x_i' \; b_i))$$

L'ensemble des indices $1..n$ est décomposé en deux sous-ensembles distincts K et $\{1..n\} - K$ tels que $i \in K$ ssi aucune variable de $x' - x''$ ne figure dans b_i. Du fait que les variables de $x' - x''$ ne figurent pas dans a, cette formule s'écrit :

$$\neg(\exists x \; a \wedge \bigwedge_{i \in K} \neg(\exists y_i x_i' \; b_i) \wedge (\exists (x' - x'') \bigwedge_{i \in \{1..n\} - K} \neg(\exists y_i x_i' \; b_i)))$$

On remarque que dans les formules $\neg(\exists y_i x_i' \; b_i)$, toutes les équations et toutes les variables sont accessibles. Par conséquence, d'après la propriété 7,

$$\mathbf{T} \models \exists (x' - x'') \bigwedge_{i \in \{1..n\} - K} \neg(\exists y_i x_i' \; b_i)$$

La formule gauche de la règle est donc équivalente à :

$$\neg(\exists x \; a \wedge \bigwedge_{i \in K} \neg(\exists y_i x_i' \; b_i))$$

En renommant les variables quantifiées pour respecter les conditions des formules de travail et, en associant aux occurrences de $\neg$ des indices appropriés, on obtient une formule de travail, qui est effectivement la formule droite de cette règle. Cette règle est donc correcte.

4.4.2.3. Justification de la règle 11

$$\neg^3 \left[\begin{array}{l} \exists x \; a \wedge q[x] \wedge \\ \neg^4 \left[\exists y \; b \wedge \bigwedge_{i=1}^{n} \neg^4 (\exists z_i \; c_i[xy]) \right] \end{array} \right] \Longrightarrow \left[\begin{array}{l} \neg^3 (\exists x \; a \wedge q[x] \wedge \neg^4 (\exists y \; b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^3 (\exists x_i y_i z_i \; c_i[x_i y_i] \wedge q[x_i]) \end{array} \right]$$

En mettant le signe $\bigwedge$ dans la portée de la troisième occurrence de $\neg$, la formule gauche devient :

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b \wedge \neg \bigvee_{i=1}^{n} (\exists z_i\ c_i)))$$

Du fait que les variables de y et les équations de b sont accessibles dans $\exists y\, b$, d'après la propriété 4, on a $\mathbf{T} \models \exists ?y\ b$. D'après la propriété 2, la formule est équivalente à :

$$\neg(\exists x\ a \wedge q \wedge \neg((\exists y\ b) \wedge \neg(\exists y\ b \wedge \bigvee_{i=1}^{n} (\exists z_i\ c_i))))$$

En distribuant le $\wedge$ sur le $\vee$ et le $\exists$ sur le $\vee$, du fait que l'ensemble des équations de b est inclus dans celui de c_i, la formule est équivalente à :

$$\neg(\exists x\ a \wedge q \wedge \neg((\exists y\ b) \wedge \neg \bigvee_{i=1}^{n} (\exists z_i y\ c_i)))$$

En faisant descendre la deuxième occurrence de $\neg$ et ensuite, en distribuant le $\wedge$ sur le $\vee$, cette formule devient :

$$\neg((\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \vee \bigvee_{i=1}^{n} (\exists x y z_i\ c_i \wedge a \wedge q))$$

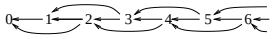
Du fait que l'ensemble des équations de a est inclus dans celui de c_i, en faisant descendre la première occurrence de $\neg$, cette formule devient :

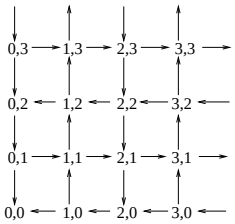
$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \wedge \bigwedge_{i=1}^{n} \neg(\exists x y z_i\ c_i \wedge q)$$

En renommant les variables quantifiées pour respecter les conditions des formules de travail et, en associant aux négations des indices appropriés, on obtient effectivement la formule droite de la règle. La correction de la règle 11 est donc prouvée. Ceci termine la preuve de la correction de l'algorithme.

5. Exemples et benchmarks

Nous avons implanté notre algorithme, en choisissant la stratégie où les sous-formules sont simplifiées de gauche à droite et en restreignant, dans la règle 11, la formule q en une conjonction de formules de travail finales. Pour être plus à l'aise, nous acceptons les occurrences de prédicats dans la formule d'entrée. Ces occurrences sont toutes remplacées par les formules de définition de prédicat avant la phase de résolution. Nous présentons dans cette partie quelques benchmarks résolvant des formules faisant intervenir de longues imbrications de quantificateurs alternés. Dans ce qui suit, nous n'exposons que la formule à résoudre, l'étude détaillée de ces exemples est présentée dans [COL 00]. Considérons les deux jeux suivants.





Jeu 1 On se donne un entier positif ou nul i et à tour de rôle on soustrait 1 ou 2 de i sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Jeu 2 On se donne un couple (i, j) d'entiers positifs ou nuls et à tour de rôle on choisit l'un des deux entiers i ou j. Suivant que l'entier choisi est impair ou pair, on augmente ou on diminue l'autre de 1 sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Soit x une position du jeu et supposons que ce soit au tour de la personne A de jouer. On dit qu'une position x est k-gagnante et on écrit $\mathit{gagnant}_k(x)$, si quelle que soit la façon de jouer de l'autre personne B, il est toujours possible que A gagne en jouant au plus k coups. En [COL 00] on montre que :

$$\mathit{gagnant}_k(x) \;\leftrightarrow\; \left[\begin{array}{l} \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \ldots \\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \mathit{faux} \qquad\qquad \underbrace{)\ldots)}_{2k} \end{array}\right] \qquad [9]$$

où $\mathit{coup}(x, y)$ signifie quelque chose de voisin de « de la position x en jouant un coup, on atteint la position y ». En descendant les négations, on obtient une imbrication de $2k$ quantificateurs alternés. On modélise les jeux dans l'algèbre des arbres $(A, \mathbf{F})$, où chaque position est représentée par un arbre. Lors de la résolution, si l'on donne en entrée la formule $\mathit{gagnant}_k(x)$, on obtient en résultat une formule qui représente toutes les positions k-gagnantes.

Considérons le jeu 1. Supposons que $\mathbf{F}$ contienne les symboles 0 et s d'arités respectives 0 et 1. Codons les entiers i par les arbres[1] $s^i(0)$. La relation *coup* est :

$$\mathit{coup}(x,y) \;\leftrightarrow\; x = s(y) \vee x = s(s(y)) \vee (\neg(x=0) \wedge \neg(\exists u\, x{=}s(u)) \wedge x{=}y)$$

Pour $\mathit{gagnant}_1(x)$, on obtient

$$\neg(\neg(\exists u\ x = s(u) \wedge u = 0) \wedge \neg(\exists u_1 u_2\ x = s(u_1) \wedge u_1 = s(u_2) \wedge u_2 = 0))$$

1. Bien entendu, $s^0(x) = x$ et $s^{i+1}(x) = s(s^i(x))$

Considérons maintenant le jeu 2. Supposons que **F** contienne les symboles 0, f, g et c, d'arités respectives 0,1,1 et 2. Codons les couples (i,j) par $c(\overline{i},\overline{j})$ avec $\overline{i} = (fg)^{\frac{i}{2}}(0)$ si i est pair et $\overline{i} = g(\overline{i-1})$ si i est impair[2]. On montre que :

$$\mathit{coup}(x,y) \;\leftrightarrow\; \mathit{transition}(x,y) \vee (\neg(\exists u\, \exists v\, x = c(u,v)) \wedge x = y)$$

avec

$$\mathit{transition}(x,y) \;\leftrightarrow\; \begin{bmatrix} \exists u\, \exists v\, \exists w \\ \begin{bmatrix} (x = c(u,v) \wedge y = c(u,w)) \vee \\ (x = c(v,u) \wedge y = c(w,u)) \end{bmatrix} \\ \wedge \\ \begin{bmatrix} (\exists i\, u = g(i) \wedge \mathit{suc}\,(v,w)) \vee \\ (\neg(\exists i\, u = g(i)) \wedge \mathit{pred}(v,w)) \end{bmatrix} \end{bmatrix}$$

$$\mathit{suc}\,(v,w) \;\leftrightarrow\; \begin{bmatrix} ((\exists j\, v = g(j)) \wedge w = f(v)) \vee \\ (\neg(\exists j\, v = g(j)) \wedge w = g(v)) \end{bmatrix}$$

$$\mathit{pred}(v,w) \;\leftrightarrow\; \begin{bmatrix} (\exists j\, v = f(j) \wedge \begin{bmatrix} (\exists k\, j = g(k) \wedge w = j) \vee \\ (\neg(\exists k\, j = g(k)) \wedge w = v) \end{bmatrix}) \vee \\ (\exists j\, v = g(j) \wedge \begin{bmatrix} (\exists k\, j = g(k) \wedge w = v) \vee \\ (\neg(\exists k\, j = g(k)) \wedge w = j) \end{bmatrix}) \vee \\ (\neg(\exists j\, v = f(j)) \wedge \neg(\exists j\, v = g(j)) \wedge \neg(v = 0) \wedge w = v) \end{bmatrix}$$

Pour la formule $\mathit{gagnant}_1\,(x)$, on obtient

$$\neg \begin{bmatrix} \neg(\exists u_1 u_2 u_3\; x = c(u_1,u_2) \wedge u_1 = g(u_2) \wedge u_2 = 0 \wedge u_3 = 0) \wedge \\ \neg(\exists u_1 u_2 u_3\; x = c(u_1,u_2) \wedge u_2 = g(u_3) \wedge u_1 = 0 \wedge u_3 = 0) \end{bmatrix}$$

Les temps d'exécution des formules $\mathit{gagnant}_k\,(x)$ sont montrés dans le tableau suivant. Les programmes sont écrits en langage C++, les benchmarks sont effectués sur un processeur Pentium II 350MHz, les temps sont donnés en milisecondes.

k	0	1	2	4	10	20	40	80
Jeu 1	0	0	10	20	300	4 270	89 870	3 841 220
Jeu 2	0	150	360	840	5 970	236 350	-	-

6. Conclusion

Nous avons présenté un algorithme de résolution de contraintes du premier ordre dans la théorie **T** des arbres finis ou infinis avec un ensemble infini de symboles fonctionnels. La résolution est effectuée par un ensemble de 11 règles de réécriture de sous-formules, qui s'adaptent bien à la réalisation. Nous avons implanté ces règles de

2. Bien entendu, $(fg)^0(x) = x$ et $(fg)^{i+1}(x) = f(g((fg)^i(x)))$

réécriture et avons effectué quelques benchmarks avec des formules faisant intervenir de longues imbrications de quantificateurs alternés. La correction de l'algorithme constitue une autre preuve de la complétude de **T** démontrée par Michael Maher. Le problème de décision de la validité d'une proposition dans la théorie **T** est démontré non-élémentaire par Sergei Vorobyov [VOR 96], c'est-à-dire que la complexité de tout algorithme qui le résout n'est pas bornée supérieurement par une tour d'exponentielles de hauteur fixe. La preuve de la terminaison de l'application des règles de réécriture illustre bien cette propriété. En effet, la fonction hyper-exponentielle α introduite dans cette preuve fournit aussi une estime de la complexité de l'algorithme dans les pires cas. Cependant malgré cette complexité proche de l'indécidabilité, nos benchmarks montrent qu'il existe des classes de problèmes, avec beaucoup de quantifications, que l'on peut résoudre en pratique. Il serait intéressant d'isoler ces classes.

7. Bibliographie

[BEN 96] BENHAMOU F., BOUVIER P., COLMERAUER A., GARETTA H., GILETTA B., MASSAT J., NARBONI G., N'DONG S., PASERO R., PIQUE J., TOURAÏVANE, VAN CANEGHEM M., VÉTILLARD E., « Le manuel de Prolog IV », PrologIA, Marseille, 1996.

[COL 82] COLMERAUER A., « Prolog and Infinite Trees », CLARK K., TARNLUND S.-A., Eds., *Logic Programming*, New York, 1982, Academic Press, p. 231-251.

[COL 83] COLMERAUER A., KANOUI H., VAN CANEGHEM M., « Prolog, Theoretical Principles and Current Trends », *Technology and Science of Informatics*, vol. 2, n° 4, 1983, North Oxford Academic, Version anglaise de la revue *TSI*, AFCET-Bordas.

[COL 84] COLMERAUER A., « Equations and Inequations on Finite and Infinite Trees », *Proceeding of the International Conference on Fifth Generation Computer Systems (FCGS-84)*, Tokyo, 1984, ICOT, p. 85-99.

[COL 90] COLMERAUER A., « An Introduction to Prolog III », *Communications of the ACM*, vol. 33, n° 7, 1990, p. 68-90.

[COL 00] COLMERAUER A., DAO T.-B.-H., « Expressiveness of Full First Order Constraints in the Algebra of Finite and Infinite Trees », *soumis à CP'2000*, 2000.

[COM 88] COMON H., « Unification et disunification : Théorie et applications », PhD thesis, Institut National Polytechnique de Grenoble, 1988.

[COU 83] COURCELLE B., « Fundamental Properties of Infinite Trees », *Theoretical Computer Science*, vol. 25, n° 2, 1983, p. 95-169.

[COU 86] COURCELLE B., « Equivalences and Transformations of Regular Systems - Applications to Program Schemes and Grammars », *Theretical Computer Science*, vol. 42, 1986, p. 1-122.

[HUE 76] HUET G., « Résolution d'équations dans les langages d'ordre $1, 2, \ldots, \omega$ », Thèse d'Etat, Université Paris 7, 1976.

[LYN 64] LYNDON R. C., *Notes on Logic*, Van Nostrand Mathematical Studies, 1964.

[MAH 88] MAHER M., « Complete Axiomatization of the Algebra of Finite, Rational and Infinite Trees », rapport, 1988, IBM - T.J. Watson Research Center.

[VOR 96] VOROBYOV S., « An Improved Lower Bound for the Elementary Theories of Trees », *Proceeding of the 13th International Conference on Automated Deduction, CADE'96*, vol. 1104 de *Springer Lecture Notes in Artificial Intelligence*, 1996, p. 275-287.